Dynamo: A Model Transition Framework for Dynamic Stability Control and Body Mass Manipulation

FEASIBILITY STUDY – FINAL TECHNICAL REPORT

Contract Number:	W91CRB-11-C-0049
Sponsoring Agency:	DARPA Defense Sciences Office (DSO)
Report Type:	Final Technical Report
Period of Performance:	January 24, 2011 – January 24, 2012
Contractor Name:	iRobot Corporation
Contractor Address:	8 Crosby Drive MS 8-1 Bedford, MA 01730
Contractor Phone:	(781) 430-3291
Contractor Email:	yamauchi@irobot.com
Principal Investigator:	Dr. Brian Yamauchi
Security Classification:	Unclassified
Distribution Statement:	Approved for public release Distribution unlimited

Sponsored by the U.S. Government under contract No. W91CRB-11-C-0049.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
NOV 2011		**00-00-2011 to 00-00-2011**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Dynamo: A Model Transition Framework For Dynamic Stability Control And Body Mass Manipulation	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **iRobot Corporation,8 Crosby Drive MS 8-1,Bedford,MA,01730**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**27**	

1 Introduction

Figure 1: Rubble from a destroyed building in Afghanistan (left), Air Force Sergeant prepares to send PackBot on mission to incident area (right)

Ground robots, such as the iRobot PackBot, have saved hundreds of lives in Iraq and Afghanistan by helping soldiers safely inspect and disarm improvised explosive devices (IEDs) (Figure 1). However, even state-of-the-art robot control systems fail to approach the adaptive, versatile mobility demonstrated by humans and animals on an everyday basis. In order to extend the applicability of ground robots to a wider range of missions, fundamental advances are needed to provide robust mobility in unstructured, dynamic, natural environments.

One of the common criticisms of currently deployed UGVs is that they are too slow. The typical standoff distance for an EOD team investigating an IED is 200-500 meters. At top speed (5.8 mph), a PackBot would take over 3 minutes to travel downrange to an IED 500 meters away. Though a defensive perimeter is setup around the EOD teams, 3 minutes is long enough to setup and deploy mortars or for snipers to take action. Reducing this time window is critical. The Robotic Systems Joint Project Office (RSJPO) has identified the need for faster UGVs as a top priority for Army and Marine Corps warfighters [Jackowski 10].

Figure 2: Italian soldier inspects IED in culvert in Afghanistan (left), iRobot PackBot retrieves pipe bomb from culvert (right)

We envision UGVs that can travel 30-45 mph or faster which would enable their use in rapid reconnaissance and high-optempo infantry assault missions. At 45 mph, a high-speed UGV could reach an IED 500 meters away in just 25 seconds. However, high-speed UGVs are difficult to

Sponsored by the U.S. Government under contract No. W91CRB-11-C-0049.

control manually, particularly on the dirt or sand surfaces that are common on the battlefield. On these surfaces, the vehicle can skid or lose traction while cornering and can become airborne from even modest terrain variation.

Another urgent warfighter need is for UGVs that can robustly negotiate rough terrain. IEDs are often emplaced in rubble or in culverts next to roads (Figure 2). These locations can be difficult for a soldier to access without putting himself at risk. Once the soldier is close enough to determine that an IED is present, he may be within the kill radius. UGVs can approach these IEDs while keeping soldiers out of harm's way, but UGVs are difficult to teleoperate over rough terrain and often have problems with rolling over or getting high-centered on obstacles. When this occurs, a solider often must walk out to rescue the robot, putting himself at risk.

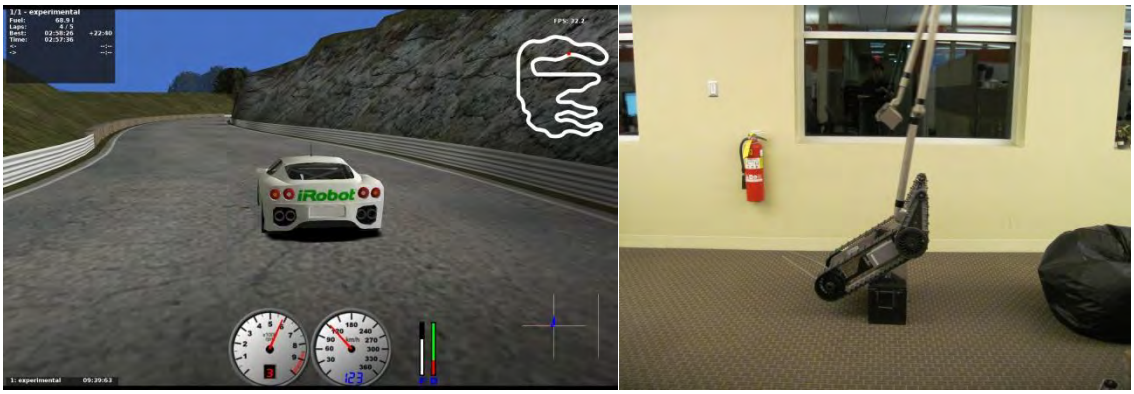

Figure 3: Simulated Ferrari controlled by MTC races on an alpine road course (left), PackBot climbs obstacle using MTC-controlled active weight shifting (right)

The Dynamo Project is developing real-time learning techniques for high-speed vehicle control and robust mobility over rough terrain. We have developed Model Transition Control (MTC), a novel framework for modeling nonlinear control problems, and Dynamic Threshold Learning (DTL), a real-time, online algorithm that learns to keep a robot within a desired control regime. During the Dynamo Feasibility Study, we have conducted research to validate the potential of our approach. In particular, we have:

- Applied DTL to learn an MTC Dynamic Stability Control (DSC) behavior for a simulated high-speed race car traveling at speeds up to 210 kph (130 mph) (Figure 3, left).
- Developed an adaptive regime classifier that can detect high-centering on a real iRobot PackBot and trigger an escape behavior.
- Applied DTL to learn an MTC rollover prevention behavior that prevents the PackBot from rolling over in rough terrain.
- Applied DTL to learn an MTC weight shifting behavior that enables the PackBot to actively adjust its center-of-gravity to climb over tall obstacles (Figure 3, right).

The results from the Feasibility Study indicate that MTC and DTL can be applied to wide range of adaptive mobility tasks, and that these techniques can be used to quickly learn the dynamic limits of both wheeled and tracked platforms in different environments. The Dynamo Project is now ready to move on to the next phase. In Phase II, we will use MTC and DTL to learn semi-autonomous control behaviors for real, high-speed, small UGVs that can travel at speeds up to 70 kph (45 mph), and we will extend our work on adaptive rough terrain mobility to more complex real-world terrain, such as rock piles and rubble piles.

Sponsored by the U.S. Government under contract No. W91CRB-11-C-0049.

2 Detailed Technical Approach

2.1 Model Transition Control (MTC)

Controlling a vehicle in natural terrain is an inherently nonlinear problem that requires adaptation to changes in the interaction between vehicle and environment. Well-understood techniques, such as Model Reference Adaptive Systems (MRAS) and Self-Tuning Regulators (STR), exist for developing adaptive controllers for linear control problems [Åström and Wittenmark, 2008], but there are no general solutions for arbitrary nonlinear control problems.

We propose Model Transition Control (MTC) as a new framework for modeling nonlinear control problems and developing nonlinear control systems. While the space of nonlinear control problems is extremely large, we can make this space more tractable through the use of domain knowledge. The key insight of MTC is that UGV control problems tend to consist of relatively linear control regimes that are linked by rapid nonlinear transitions. This insight motivates the *state-action map* (Figure 4), which takes the current vehicle state and proposed vehicle action as inputs and outputs the regime that will result if the proposed action is taken. For example, the state-action map may say that if you're driving at high speed, and you turn the steering wheel hard to the right and slam on the brakes, then you will end up in the oversteer regime. At the same time, it may also say that you can turn less sharply and brake more gradually and remain in the normal grip regime.

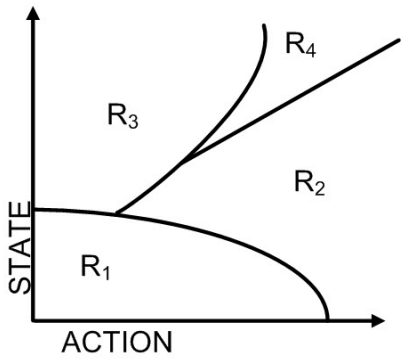

Figure 4: State-action map

For a concrete example, consider the dynamic transition that occurs when a vehicle's brakes lock up. Before the brakes lock up, vehicle deceleration can be approximated as a linear function of braking force at the wheels. After the brakes lock up, vehicle deceleration is no longer a function of braking force. Instead, deceleration becomes a function of vehicle weight, velocity, and the sliding coefficient of friction between the tires and the ground.

Anti-lock braking systems (ABS) handle this by monitoring the rotational speed of each wheel and detecting when one of those wheels starts moving much slower than the others, indicating that it is about to lock up. ABS then reduces the braking force at that individual wheel, allowing it to start moving faster. After the wheel starts moving, ABS increases the braking force again until it senses that the wheel is about to lock up again. ABS allows the average driver to stop a car in a relatively short distance by simply pressing the brake hard and holding it down. However, professional race drivers can achieve even better performance through a technique known as *threshold braking*. In threshold braking, the driver learns the point at which the brakes start to lock up and then applies slightly less pressure. This provides the maximum braking force to the

Sponsored by the U.S. Government under contract No. W91CRB-11-C-0049.

wheels, as opposed to ABS which alternates between maximum braking force (no lock up) and reduced braking force (lock up).

Like a professional race driver, MTC DSC will learn the point at which the brakes will lock up for a given vehicle state. A simplified two-dimensional version of the corresponding state-action map (SAM) is shown in Figure 5. In reality, the state-action map would have multiple dimensions for both vehicle state and potential actions. This map shows that small amounts of braking force will never cause the brakes to lock up, so the vehicle will remain in the normal grip regime. The map also shows that, at low speeds, full braking can be applied without causing lock up. At high speeds, hard braking will result in a skid, as indicated by the red region showing the skid regime in the map.

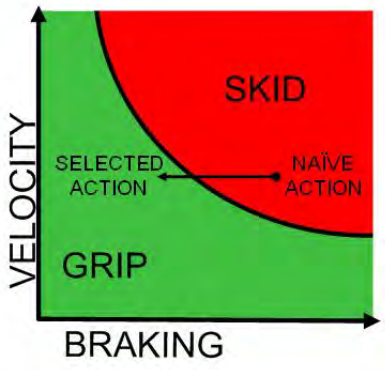

Figure 5: State-action map for braking

Given the vehicle state and a naïve action that might be taken by an inexperienced driver or generated by a simple PID control loop, we can use the state-action map to identify the resulting control regime. If the line corresponding to the current vehicle slate intersects the region corresponding to the desired control regime, we can find an action along that line that will result in the desired control regime (Figure 5). In practice, the current vehicle state will be represented by an n-dimensional manifold corresponding to the n degrees-of-freedom in the control space. For low-dimensional spaces, an exhaustive search can be used to find the action closest to the naïve action that will put the vehicle in the desired control regime. For high-dimensional spaces, exhaustive search is computationally infeasible, so a Monte Carlo approach will be used instead, with a large number of actions being generated randomly, and the action that is closest in Euclidean space to the naïve action will be selected.

Figure 6 shows a block diagram for an MTC controller using Dynamic Threshold Learning (DTL). An exploratory controller provides raw commands based on sensor inputs, which are used to explore the state-action space. For example, this could be a PID controller or a simple set of reactive rules. Alternately, a human operator can provide teleoperation inputs that serve the role of the exploratory controller. The exploratory controller provides aggressive responses that are likely to result in both desirable and undesirable control regimes. As the MTC controller learns the SAM, it gradually limits these responses to remain within the desired control regime.

If the DTL module predicts that the raw command output by the exploratory controller will result in an undesirable control regime, then the DTL will modify the output in one of two ways. First, it can modify the control law parameters in the exploratory controller so that it will only generate outputs that remain within the desired control regime. Second, it can apply explicit limits to the command outputs via the arbiter.

Sponsored by the U.S. Government under contract No. W91CRB-11-C-0049.

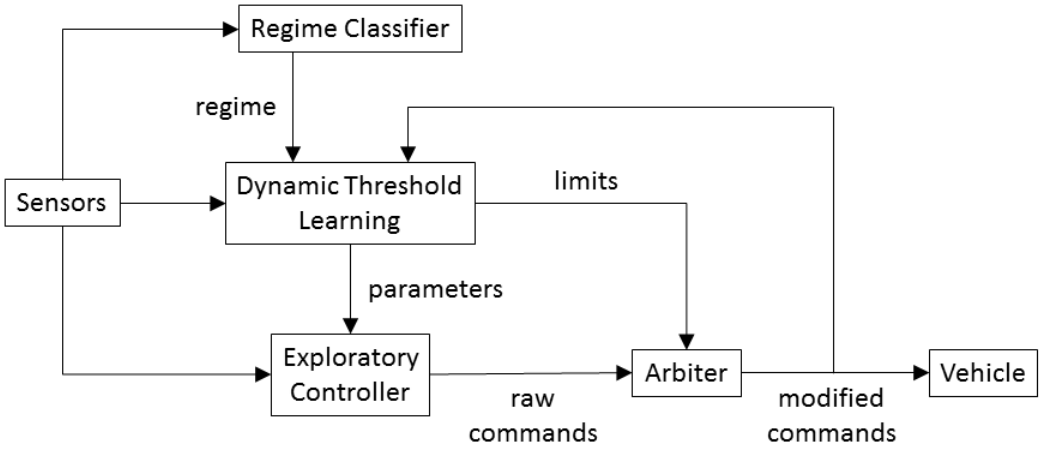

Figure 6: Block diagram for MTC controller

The net result is that an MTC-controlled robot will initially behave very aggressively, and this will result in the robot experiencing undesirable control regimes. DTL will quickly learn which state-action combinations lead to undesirable control regimes and will modify the control system so that it no longer directs the robot into these regimes. After a brief learning period, the MTC-based control system will be able to keep the robot within the desired control regime regardless of dynamic changes in the environmental inputs.

2.2 Dynamic Transition Learning (DTL)

The MTC framework is compatible with a wide variety of learning techniques. We have developed a novel learning algorithm that we call dynamic transition learning (DTL), which makes use of domain knowledge related to vehicle dynamics to speed up the learning process. Transitions between vehicle control regimes tend to occur when some physical threshold (e.g. tire adhesion) is exceeded. This allows us to generalize from a single observation to all other points in the state-action space that exceed the same threshold in the same way. By making this assumption of monotonicity in regime transition thresholds, we sacrifice some generality in terms of the functions that can be learned, but in return, we gain very fast learning. A key feature of our approach is that all learning takes place online as the vehicle moves through the world. Initially, the vehicle's motions are controlled by the exploratory controller, but DTL continuously learns regime transition dynamics as the vehicle moves through the world and continuously improves vehicle performance via MTC.

We assume a bounded state-action space where each *state* can vary between $state_{min}$ and $state_{max}$ and each *action* can vary between $action_{min}$ and $action_{max}$, and we define an anchor point A_{REGIME} in state-action space for each *REGIME*. This anchor point corresponds to the combined state and action that is most likely be part of the corresponding control regime. In the case of braking, skidding is most likely to occur when the vehicle is traveling at the maximum speed and maximum brake effort is applied. So the anchor point for the skid control regime is defined by:

$$A_{SKID}=(v_{max}, brake_{max})$$

We define a regime identification function *IDENT* such that:

$$IDENT(S_i,A_i) = REGIME$$

Where S_i is a state vector, A_i is an action vector, and *REGIME* is the control regime that applies when action A_i is executed in state S_i.

Sponsored by the U.S. Government under contract No. W91CRB-11-C-0049.

Given that $IDENT(S_i,A_i) = REGIME$, then for any other point in state-action space (S_j,A_j), we assume that $IDENT(S_j,A_j) = REGIME$, if for each state variable *state* in S_j:

$$state_i \leq state_j \leq A_{REGIME}(state) \text{ or } state_i \geq state_j \geq A_{REGIME}(state)$$

And for each action variable action in A_j:

$$action_i \leq action_j \leq A_{REGIME}(action) \text{ or } action_i \geq action_j \geq A_{REGIME}(action)$$

Where $A_{REGIME}(state)$ is the value of state variable *state* for the *REGIME* anchor point, and $A_{REGIME}(action)$ is the value of action variable *action* for the *REGIME* anchor point. In other words, if the new state-action point is closer to the anchor point along *every* axis, we can infer that it corresponds to the same regime. If (S_j,A_j) is not within the corresponding region for any observation, *IDENT* returns the default regime. This allows DTL to rapidly converge to an approximate model of the system dynamics with a small number of observations.

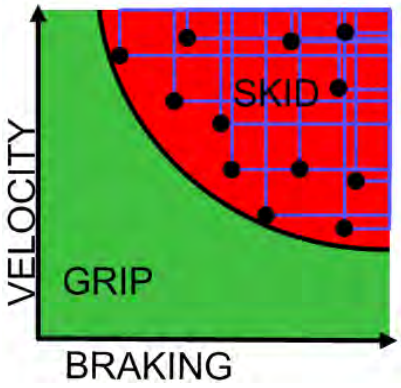

Figure 7: Using DTL to learn SKID regime boundaries from a small number of observations (black circles)

Figure 7 shows how a handful of observations can be used to create a reasonable model of the SKID regime in the braking example. DTL constructs a model of the boundaries between control regimes with a relatively small number of observations. Points near regime boundaries are particularly useful for constructing this model, and this suggests the exploratory controller should drive the vehicle close to its limits to speed up learning. In higher-dimensional spaces, more observations will be necessary, but we believe that DTL will scale well since each observation can update the regime information for a large volume of the state-action space.

3 Simulated High-Speed Vehicle Control

For our first experiments with DTL, we used The Open Race Car Simulator (TORCS), an open-source high-speed vehicle simulator. Source code for TORCS is available online at http://torcs.sourceforge.net. We chose TORCS because it provides physical modeling of high-speed wheeled vehicles, including tire-surface interaction for different surface materials and geometries, aerodynamic drag, and handling parameters for different vehicles. TORCS also provides an easy-to-use interface for implementing autonomous driving behaviors and linking these behaviors with the simulator.

We developed an MTC-based dynamic stability control (DSC) system for simulated TORCS vehicles to prevent understeer and oversteer. Conventional DSC systems are only capable of reacting to understeer and oversteer once they occur. These systems detect the difference between the actual and commanded yaw rates and then apply brakes at individual wheels and/or reduce the

Sponsored by the U.S. Government under contract No. W91CRB-11-C-0049.

throttle to reduce understeer/oversteer. Our MTC-based DSC system predicts which state-action combinations will result in understeer or oversteer. It then modifies the steering and throttle commands to prevent understeer and oversteer from occurring in the first place.

3.1 Exploratory Controller

The exploratory controller attempts to steer the car toward the center of the current track segment while maintaining a desired speed. The steering control rule is:

$$\theta_{steer} = (\theta_{track} - \theta_{car}) - \frac{y_{car}}{w_{track}}$$

where θ_{steer} is the steering command, θ_{track} is the track orientation in world coordinates, θ_{car} is the vehicle heading in world coordinates, y_{car} is the lateral position of the car relative to the current track segment, and w_{track} is the width of the current track segment.

The speed control rule is:

$$a = k_v(v_{desired} - v_{current})$$

where a is the acceleration command, k_v is the proportional control gain, $v_{desired}$ is the desired velocity, and $v_{current}$ is the current velocity. Initially, $v_{desired}$ is set to the maximum speed of the vehicle.

3.2 Preventing Understeer

We define an understeer threshold that is a function of speed, turn radius, and the friction coefficient of the track surface. This threshold specifies the maximum speed that the vehicle can drive for a given turn radius without risking understeer.

The maximum traction available is given by:

$$F = ma = \mu g m \rightarrow a = \mu g$$

where F is the force applied by the tires, m is the mass of the vehicle, μ is the coefficient of friction, and g is the acceleration due to gravity. The lateral acceleration during a turn is:

$$a = \frac{v^2}{r}$$

where a is the lateral acceleration, v is the vehicle velocity, and r is the turn radius. We predict understeer when the lateral acceleration exceeds the maximum traction:

$$\frac{v^2}{r} > \mu g = T_{understeer}$$

where $T_{understeer}$ is the understeer threshold, which is initialized to its maximum value.

Sponsored by the U.S. Government under contract No. W91CRB-11-C-0049.

The task of learning the SAM for the understeer prevention behavior is equivalent to learning $T_{understeer}$, assuming the regime boundary is linear in the space defined by the state variable for the track radius and the action variable for velocity. In this SAM, the anchor point for the GRIP regime is at maximum *r* and zero *v*. The anchor point for the UNDERSTEER regime is at minimum *r* and maximum *v*.

Understeer is detected by the regime classifier when the absolute yaw rate $\dot{\theta}_{current}$ is less than the absolute steering command θ_{steer} multiplied by a constant $k_{understeer}$ that is less than 1:

$$|\dot{\theta}_{current}| < k_{understeer}|\theta_{steer}|$$

If understeer is detected, but understeer is not predicted, then $T_{understeer}$ is reduced to the current lateral acceleration:

$$T_{understeer} = \frac{v^2}{r}$$

If DTL predicts that the current desired velocity will exceed the understeer threshold for the turn radius of the current track segment,

$$\frac{v_{desired}^2}{r} > T_{understeer}$$

then DTL reduces the desired velocity to a value that is not predicted to cause understeer:

$$v_{desired} = \sqrt{rT_{understeer}}$$

3.3 Preventing Oversteer

We define an oversteer threshold as a function of steering and acceleration inputs. This threshold represents the maximum combination of steering and throttle inputs that can be commanded without triggering oversteer.

We predict oversteer when

$$k_{steer}|\theta_{steer}| + k_{accel}a > T_{oversteer}$$

where k_{steer} and k_{accel} are fixed coefficients for weighting the steering and acceleration inputs and $T_{oversteer}$ is the oversteer threshold, which is initialized to its maximum value.

The regime classifier detects oversteer when the absolute yaw rate $\dot{\theta}_{current}$ is greater than the absolute steering command θ_{steer} multiplied by a constant $k_{oversteer}$ that is greater than 1:

$$|\dot{\theta}_{current}| > k_{oversteer}|\theta_{steer}|$$

Sponsored by the U.S. Government under contract No. W91CRB-11-C-0049.

If oversteer is detected, but not predicted, then we reduce the oversteer threshold based on the current steering and acceleration commands:

$$T_{oversteer} = k_{steer}|\theta_{steer}| + k_{accel}a$$

If the steering and acceleration commands proposed by the exploratory controller are predicted to exceed the oversteer threshold:

$$k_{steer}|\theta_{steer}| + k_{accel}a + k_{margin} > T_{oversteer}$$

where k_{margin} is a fixed safety margin, then we reduce the acceleration command (if possible) to a value that is not predicted to cause oversteer:

$$a_{mod} = \min\left(\frac{T_{oversteer} - k_{steer}|\theta_{steer}| - k_{margin}}{k_{accel}}, 0\right)$$

where a_{mod} is the modified acceleration command. We do not reduce the steering command, because doing so would be the equivalent of inducing understeer and would reduce the maneuverability of the vehicle.

3.4 Experimental Results

We tested the MTC DSC controller on a wide range of track types including speedways with long straights and sweeping turns that allow high speeds (Figure 8); complex road courses with tight corners that allow moderate speeds (Figure 9); and small dirt tracks with tight corners that require slow speeds (Figure 10). We experimented with tracks containing both asphalt (high friction) and dirt (low friction) surfaces. For these tracks, DTL maintained separate $T_{understeer}$ and $T_{oversteer}$ parameters for asphalt and dirt learned the correct parameters for the current terrain type. On a real vehicle, this would require a terrain classification system using sensors such as vision.

In our initial experiments, we discovered that after the car began to understeer or oversteer, it would often end up switching rapidly between GRIP, UNDERSTEER, and OVERSTEER domains. Consider a driver who begins to spin out and then countersteers in the opposite direction, the car may end up snapping back and forth as the tires skid and grab intermittently. Once an understeer or oversteer event begins, the transitions between GRIP and UNDERSTEER and OVERSTEER occur at much lower speeds, steering angles, and acceleration inputs. As a result, the initial MTC DSC controller learned overly conservative regime limits and drove much more slowly than necessary.

Sponsored by the U.S. Government under contract No. W91CRB-11-C-0049.

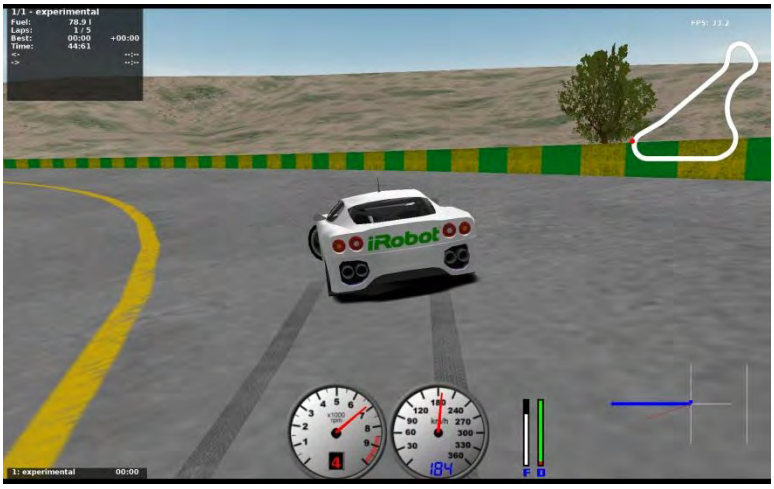

Figure 8: MTC controller driving simulated Ferrari on speedway

Figure 9: MTC controller driving simulated Ferrari on alpine road course

Figure 10: MTC controller driving simulated Ferrari on dirt track

Sponsored by the U.S. Government under contract No. W91CRB-11-C-0049.

In order to prevent this, we added an event timer, t_{event}. When $t_{event} = 0$, and understeer or oversteer is detected, the transition is learned by DTL, and t_{event} is set equal to t_{max}. At every time step, if $t_{event} > 0$, then t_{event} decremented by 1. If $t_{event} > 0$ and understeer or oversteer is detected, the event is ignored by DTL, but the event timer is reset to t_{max}.

The result is that for a brief period of time, DTL ignores understeer and oversteer detections to avoid learning false positive information. A tradeoff exists between a low value of t_{max}, which will reduce the learning time, but increase false positives, and a high value of t_{max}, which will reduce false positives, but also increase learning time and increase the number of collisions. The TORCS simulator models collision damage, and if the car suffers too much damage, the simulation run is ended. So, if the value of t_{max} is set too high, the car may fail to finish the race due to collisions.

The TORCS simulator package includes 34 tracks: 17 road tracks, 8 dirt tracks, and 9 oval tracks. We tested the MTC DTL controller on all 34 tracks, and we compared its performance with the reactive controller provided in Chapters 1-3 of Bernhard Wymann's TORCS Robot Tutorial available at http://www.berniw.org/ (but without the traction control and ABS functions in Chapter 3.6). We also compared the performance of the MTC DSC controller with two different values of t_{max}, 100 timesteps (2 seconds) and 500 timesteps (10 seconds).

The reactive controller differs from the MTC exploratory controller in knowing the friction coefficient of each track segment in advance and then commanding the maximum speed that it knows will not cause loss of traction. Thus, the reactive controller makes use of more a priori knowledge about the environment than is available to the MTC DTL controller. The reactive controller is also more conservative than the MTC DTL controller, even after learning, as the MTC DTL controller will allow some wheel slip as long as the understeer and oversteer thresholds are not exceeded. Table 1 shows the lap times for reactive controller on all tracks.

The key results from our experiments are:

1) DTL is able to successfully learn thresholds for understeer and oversteer prevention across a wide range of track geometries and surface types.
2) DTL is able to learn these thresholds very quickly, often within a single lap.
3) In most cases, the MTC DSC controller can outperform the reactive controller, despite the fact that the reactive controller is provided with a prior information about the track characteristics that DTL has to learn through experience.
4) In some cases, DTL will learn overly conservative thresholds from false positives that result in reduced performance.
5) In other cases, DTL may not learn quickly enough, resulting in excessive damage to the vehicle.
6) The maximum event time t_{max} parameter controls the tradeoff between false positive frequency and learning time. Low values of t_{max} result in many false positives but rapid learning. High values of t_{max} result in fewer false positives but slower learning.

Table 2 shows the results of our experiments with $t_{max} = 500$. All times are in seconds. We measured the lap time for the MTC DSC controller on the first lap after learning had converged. For a fair comparison, we compared these lap times to the second lap time for the reactive controller (since this allowed the reactive controller to start the measured lap at speed, as with the MTC DSC controller).

Sponsored by the U.S. Government under contract No. W91CRB-11-C-0049.

Table 1: Lap times for reactive controller (seconds)

Ferrari 360 Modena	Reactive
	Time for Lap 2
ROAD TRACKS	
Alpine 1	203.18
E-Track 1	96.13
E-Track 2	110.15
E-Track 3	129.71
E-Track 4	144.69
E-Track 6	130.97
E-Road	105.82
CG Speedway 1	57.98
CG Track 2	80.11
CG Track 3	88.72
Olethros Road 1	153.52
Ruudskogen	87.87
Spring	663.41
Street 1	113.71
Wheel 1	116.3
Wheel 2	158.35
Aalborg	161.61
DIRT TRACKS	
Dirt 1	47.14
Dirt 2	68.97
Dirt 3	79.5
Dirt 4	120.83
Dirt 5	47.04
Dirt 6	130.16
Mixed 1	66.42
Mixed 2	92.57
OVAL TRACKS	
A-Speedway	46.49
B-Speedway	62.65
C-Speedway	52.99
D-Speedway	52.6
E-Speedway	64.31
E-Track 5	46.04
F-Speedway	58.5
G-Speedway	49.63
Michigan Speedway	48

In these experiments, the MTC DSC controller was able to successfully learn to prevent understeer and oversteer on 16 out 17 road tracks and 3 out of 8 dirt tracks. The MTC DSC controller was also able to complete all 9 oval tracks successfully, though learning was only needed on 2 tracks.

Sponsored by the U.S. Government under contract No. W91CRB-11-C-0049.

Table 2: Lap times for MTC DSC Controller with t_{max} = 500
(DNF = Did Not Finish)

Reactive		MTC DSC Controller t_{max} = 500				
	Time for Lap 2	Laps Required for Learning	Time for Lap 2	Time for First Lap After Learning	Time vs. Reactive	% Change
ROAD TRACKS						
Ferrari 360 Modena	203.81	2	203.18	187.25	-15.93	-7.84
Dirt track 1	90.13	1	96.13	102.31	6.18	6.43
Dirt track 2	110.15	2	110.15	91.31	-18.84	-17.10
Dirt track 3	129.71	2	129.71	116.83	-12.88	-9.93
Dirt track 4	147.69	1	144.69	125.87	-18.82	-13.01
Dirt track 6	130.97	1	130.97	105.39	-25.58	-19.53
Dirt Road	105.82	1	105.82	165.52	59.70	56.42
Chevy Speedway 1	57.96	3	57.98	47.55	-10.43	-17.99
Dirt track 2	80.11	1	80.11	63.17	-16.94	-21.15
Dirt track 3	88.72	1	88.72	88.61	-0.11	-0.12
Infineon Road 1	153.52	2	153.52	255.99	102.47	66.75
Road Skogen	87.85	1	87.87	79.21	-8.66	-9.86
Spring	663.41	1	663.41	652.02	-11.39	-1.72
Street 1	113.71	1	113.71	97.62	-16.09	-14.15
Wheel 1	116.31	1	116.3	225.48	109.18	93.88
Wheel 2	158.35	1	158.35	137.31	-21.04	-13.29
Autobahn	161.61	0	161.61	DNF	N/A	N/A
DIRT TRACKS						
Dirt 1	47.14	3	47.14	DNF	N/A	N/A
Dirt 2	68.97	1	68.97	DNF	N/A	N/A
Dirt 3	79.5	5	79.5	69.88	-9.62	-12.10
Dirt 4	120.83	1	120.83	99.48	-21.35	-17.67
Dirt 5	47.04	3	47.04	DNF	N/A	N/A
Dirt 6	130.16	2	130.16	114.17	-15.99	-12.28
Mud 1	66.42	1	66.42	DNF	N/A	N/A
Mud 2	92.57	1	92.57	DNF	N/A	N/A
OVAL TRACKS						
A-Speedway	46.49	2	46.49	36.7	-9.79	-21.06
B-Speedway	62.65	0	62.65	46.62	-16.03	-25.59
C-Speedway	52.99	0	52.99	38.89	-14.10	-26.61
D-Speedway	52.6	0	52.6	40.02	-12.58	-23.92
E-Speedway	64.31	0	64.31	47.68	-16.63	-25.86
Dirt track 5	46.04	1	46.04	33.83	-12.21	-26.52
F-Speedway	58.5	0	58.5	43.29	-15.21	-26.00
G-Speedway	49.63	0	49.63	35.37	-14.26	-28.73
Michigan Speedway	48	0	48	36.83	-11.17	-23.27

On the tracks where the MTC DSC controller was successful, learning typically converged in 3 or less laps. On many tracks, the MTC DSC controller was able to learn how to prevent understeer and oversteer in less than a single lap. On most of the oval tracks, no learning was necessary, because the tracks were sufficiently banked to allow the cars to drive flat out for the entire track. On oval tracks, the exploratory controller was able to outperform the reactive controller even without learning, because it allowed the car to slide in the corners, while the reactive controller did not.

The MTC DSC controller demonstrated significantly better performance (on those tracks where it was successful) than the reactive controller. On 24 out of 28 tracks, the MTC DSC controller was able to lap the track faster than the reactive controller, despite the additional a priori knowledge available to the reactive controller. On average, the MTC DSC controller was able to complete these 24 tracks in 17% less time than required by the reactive controller.

Sponsored by the U.S. Government under contract No. W91CRB-11-C-0049.

Table 3: Lap times for MTC DSC Controller with t_{max} = 100
(DNF = Did Not Finish, * = race terminated)

Ferrari 360 Modena	MTC DSC Controller t_{max} = 100				
	Laps Required to Learn	Time for Lap 2	Time for First Lap After Learning	Time vs. Reactive	% Change
ROAD TRACKS					
Alpine 1	12	03.18	182.09	-21.09	-10.38
E-Track 1	1	96.13	195.83	99.70	103.71
E-Track 2	11	10.15	366.32	256.17	232.56
E-Track 3	11	29.71	114.75	-14.96	-11.53
E-Track 4	11	44.69	125.87	-18.82	-13.01
E-Track 6	11	30.97	110.47	-20.50	-15.65
E-Road	11	05.82	291.51	185.69	175.48
CG Speedway 1	1	57.98	46.37	-11.61	-20.02
CG Track 2	1	80.11	63.17	-16.94	-21.15
CG Track 3	1	88.72	364.17	275.45	310.47
Olethros Road 1	11	53.52	173.66	20.14	13.12
Ruudskogen	1*	87.87	420	332.13	377.98
Spring	1*	663.41	1200	536.59	80.88
Street 1	11	13.71	97.61	-16.10	-14.16
Wheel 1	1*	116.3	480	363.70	312.73
Wheel 2	11	58.35	137.31	-21.04	-13.29
Aalborg	DNF	161.61	DNF	N/A	N/A
DIRT TRACKS					
Dirt 1	1	47.14	41.42	-5.72	-12.13
Dirt 2	2	68.97	96.6	27.63	40.06
Dirt 3	1	79.5	116.42	36.92	46.44
Dirt 4	11	20.83	99.48	-21.35	-17.67
Dirt 5	1	47.04	45.36	-1.68	-3.57
Dirt 6	11	30.16	113.97	-16.19	-12.44
Mixed 1	1	66.42	53.93	-12.49	-18.80
Mixed 2	1	92.57	75.54	-17.03	-18.40
OVAL TRACKS					
A-Speedway	2	46.49	36.7	-9.79	-21.06
B-Speedway	0	62.65	46.62	-16.03	-25.59
C-Speedway	0	52.99	38.89	-14.10	-26.61
D-Speedway	0	52.6	40.02	-12.58	-23.92
E-Speedway	0	64.31	47.68	-16.63	-25.86
E-Track 5	0	46.04	33.83	-12.21	-26.52
F-Speedway	0	58.5	43.29	-15.21	-26.00
G-Speedway	0	49.63	35.37	-14.26	-28.73
Michigan Speedway	0	48	36.83	-11.17	-23.27

However, in the 4 out of 28 tracks where the MTC DSC controller lost to the reactive controller, it required an average of 56% more time. The reason is that for three of these tracks – E-Road, Ruudskogen, and Wheel 1 – the MTC DSC controller learned overly conservative thresholds and completed these tracks at much slower speeds.

The MTC DSC controller failed to finish 6 out of the 34 tracks. On the one road track, Aalborg, the initial straightaway led directly into a sharp corner (Turn 1) with a stone wall, so the car crashed into the wall at full speed and sustained sufficient damage to end the race. On the 5 dirt tracks, the car was almost continuously understeering or oversteering, so the event timer was rarely reset, and DTL was unable to learn the correct thresholds. Eventually sufficient collision damage accumulated to end the race.

Table 3 shows the results from the experiments with t_{max} = 100. With the reduced event time, DTL tended to learn more quickly, but also more conservatively. As a result, the MTC DSC controller was able to learn how to prevent understeer and oversteer on 33 out 34 tracks. The one exception was Aalborg, where the high-speed collision on Turn 1 ended the race immediately.

Sponsored by the U.S. Government under contract No. W91CRB-11-C-0049.

The cost of this greatly improved robustness was reduced performance on some tracks, due to the more conservative thresholds learned for understeer and oversteer. However, the MTC DSC controller was still able to outperform the reactive controller on 8 out of 17 road tracks, 6 out of 8 of the dirt tracks, and all 9 oval tracks. On average, the MTC DSC controller was able to complete these 23 tracks in 15% less time than the reactive controller.

On the 10 tracks where MTC DSC controller lost to the reactive controller, it required an average of 213% more time to complete these laps. This included three tracks where the race was terminated due to lack of progress. On these tracks, the MTC DSC controller learned extremely low thresholds, and drove extremely slowly. This occurred because the event timer expired while an understeer or oversteer event was still in progress. For example, the car would snap between oversteer and understeer at low speeds and at some points during this event, the steering wheel would be pointed nearly straight and acceleration was near zero. As a result, DTL could learn wrongly that understeer and oversteer could be trigger at low speeds with small steering and acceleration inputs.

One conclusion we drew from these results is that DTL needs the capacity for unlearning false positives as well as a means for testing learning thresholds. One approach would be to probabilistically attempt to exceed learned thresholds by a small amount. As a result, these thresholds would slowly increase over time as long as the controller did not detect any understeer or oversteer events.

An alternate approach would be to modify the exploratory controller so it starts with a very conservative strategy and gradually becomes more aggressive until undesirable control regimes are encountered. This would have the additional benefit of reducing the overall number of collisions and damage suffered by the vehicle. However, this could come at the cost of increased learning time.

We plan to investigate both unlearning and the alternate exploratory controller strategy during Phase II of the Dynamo Project.

We also briefly experimented with other vehicle types, including a NASCAR stock car, a Subaru WRX STI rally car, and a dune buggy. MTC DTL worked equally well on these other vehicles to learn understeer and oversteer thresholds.

4 PackBot Rollover Prevention

For our second task, we developed an MTC-based controller to prevent rollover using a real iRobot PackBot. The SAM for this task has a state variable for the robot's pitch angle and an action variable for the robot's speed. The anchor point for the NORMAL regime is at zero pitch and speed. The anchor point for the ROLLOVER regime is at maximum pitch and speed.

These experiments were performed using stock PackBot hardware. The PackBot's pitch and roll angles were determined using the robot's standard onboard fluid pitch-roll sensor. The PackBot's speed was determined using its wheel encoders.

The role of the exploratory controller is performed by a human operator. Initially, translation and rotation commands are sent unchanged to the robot. As the operator explores the space of robot actions, DTL observes which combinations of pitch and speed result in rollover and learns the boundaries between the NORMAL and ROLLOVER control regimes.

Sponsored by the U.S. Government under contract No. W91CRB-11-C-0049.

The regime classifier identifies the current regime based on the output of the roll sensor:

$$REGIME(\varphi) = \begin{cases} NORMAL & if\ \varphi \leq 90° \\ ROLLOVER & if\ \varphi > 90° \end{cases}$$

where φ is the robot's roll angle. This classifier would work equally well using the pitch angle.

DTL learns to predict rollover when the pitch angle exceeds a threshold function of velocity:

$$\theta \geq T_{rollover}(v_{current})$$

where θ is the robot's pitch angle (0° = level, 90° = vertical), $T_{oversteer}$ is the rollover threshold function, and v is the robot's velocity.

DTL maintains a history of recent robot state-action pairs.

$$H = \{(\theta_0, v_0), (\theta_1, v_1) \ldots (\theta_n, v_n)\}$$

where H is the state-action history, θ_i is the robot's pitch angle at i timesteps prior to the present and v_i is the robot's velocity at i timesteps prior to the present and n is the number of entries in the history. In our experiments, the timesteps were 0.1 seconds long and the history contained the most recent 5 state-action values.

Figure 11: PackBot attempts to climb a tall obstacle and begins to roll over backwards

If the robot rolls over and rollover was not predicted, then for each state-action pair in the history, DTL reduces the pitch threshold for the velocity and all greater velocities:

$$\forall(\theta_i, v_i) \in \mathrm{H}: \forall v \geq v_i: T_{rollover}(v_i) = min\big(\theta_i, T_{rollover}(v_i)\big)$$

Sponsored by the U.S. Government under contract No. W91CRB-11-C-0049.

If the current pitch angle exceeds the pitch threshold for the speed commanded by the operator

$$\theta_{pitch} \geq T_{rollover}(v_{desired})$$

where $v_{desired}$ is the desired velocity sent by the operator, then the DTL module will reduce the speed command sent to the robot to the maximum speed that is not predicted to cause rollover:

$$v_{command} = max(v) \mid \theta_{pitch} < T_{rollover}(\mathrm{v})$$

where $v_{command}$ is the velocity command that is sent to the robot.

In each trial, we drove the PackBot forward at a fixed speed onto a tall obstacle (Figure 11). As the robot drove forward, it reached a point where it rolled over backward. DTL then learned each state-action pair in the history that led up to the rollover event and lowered the pitch thresholds for the corresponding robot velocities. We repeated this procedure at three different speeds (0.25 m/s, 0.5 m/s, and 1.0 m/s). DTL was able to learn how to prevent rollover at 0.25 m/s after just 2 trials. The thresholds learned for 0.25 m/s were also sufficient to prevent rollover at 0.5 m/s but not 1.0 m/s. After 5 more trials, DTL learned thresholds for preventing rollover at all speeds up to 1.0 m/s.

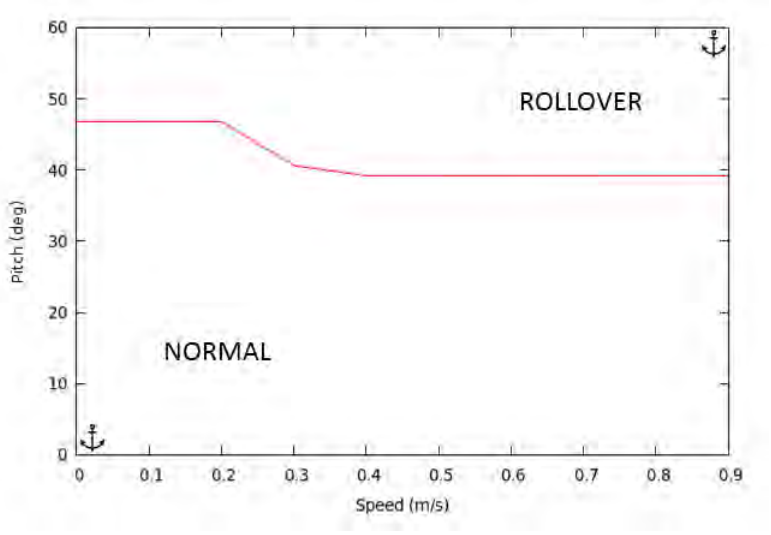

Figure 12: SAM learned by DTL for the rollover prevention task. The NORMAL regime anchor point is at zero pitch and speed. The ROLLOVER regime anchor point is at maximum pitch and speed.

Figure 12 shows the SAM learned by DTL during a typical rollover prevention learning experiment. This SAM indicates that if the robot ever exceeds a pitch angle of 47°, it should stop moving forward. However, if the pitch angle is between 39° and 47°, it can move forward safely as long as the speed is below the threshold function.

5 PackBot High-Centering Prevention

For our third task, we developed a regime classifier that learned to detect when the PackBot was high-centered on an obstacle. High-centering occurs when the PackBot drives onto an obstacle;

Sponsored by the U.S. Government under contract No. W91CRB-11-C-0049.

its momentum carries it on top of the obstacle; and the obstacle is tall enough to lift the PackBot off its treads (Figure 13, left). Once the PackBot is high-centered, it can no longer move. However, if the PackBot can detect that it is high-centered, it can rotate its flippers down to contact the ground, and then drive off the obstacle (Figure 13, right).

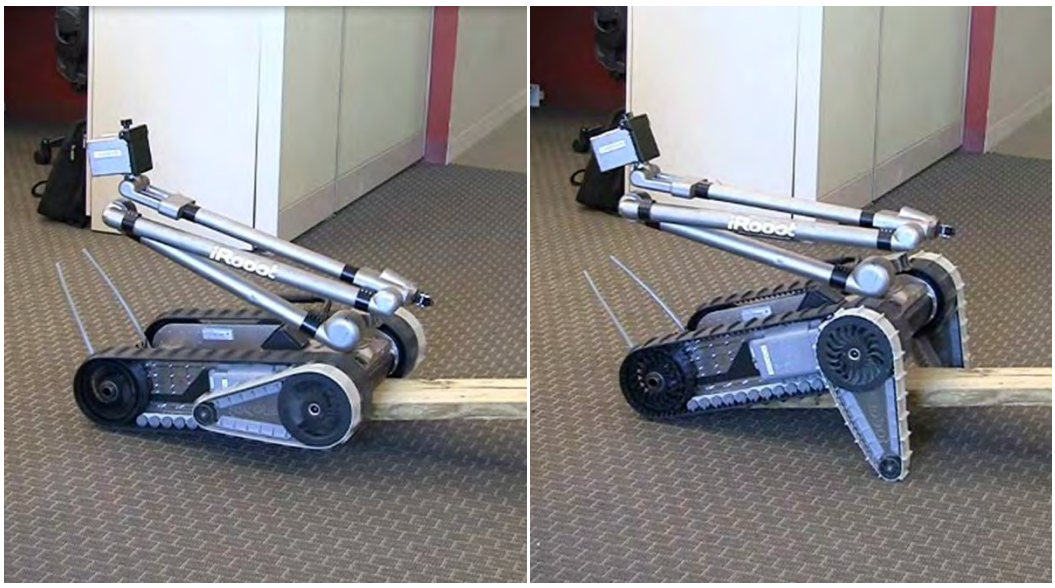

Figure 13: PackBot becomes high-centered on obstacle and spins its tracks in the air (left), High-centering prevention behavior uses flippers to free PackBot (right)

High-centering commonly occurs in the field when the PackBot is driving over rough terrain. The ability to prevent high-centering is one of the capabilities requested by the warfighters.

To detect high-centering, our regime classifier learns the amount of vibration that occurs when the robot is moving forward (NORMAL control regime) versus when it is high-centered and spinning its tracks in the air (HIGH-CENTERED control regime). The average vibration amplitude tends to be higher when the robot is moving then when it is high-centered. This is especially true when moving over rough terrain, but even on a flat, indoor, carpeted surface, we were able to detect the difference in amplitude between forward movement and spinning the tracks in the air.

The vibration is measured using the PackBot's onboard fluid pitch-roll sensor and is measured in terms of the rate of change in the robot's pitch angle. During learning, we drove the robot forward at a set of speeds ranging from 0.0 to 0.5 m/s at 0.1 m/s increments, driving at each speed for 5 seconds. The pitch angle was sampled at 10 Hz, and for each velocity v, the difference between each pitch angle $\theta_{v,t}$ and the previous pitch angle $\theta_{v,t-1}$ was stored as $\Delta\theta_{v,t}$:

$$\Delta\theta_{v,t} = \theta_{v,t} - \theta_{v,t-1}$$

The average vibration amplitude for each velocity v was then computed as the average of these differences, where n is the number of stored difference values for each velocity:

$$\varphi(v) = \frac{\sum_{t=1}^{n} \Delta\theta_{v,t}}{n}$$

We linearly interpolate this function for velocities other than ones observed during learning.

Sponsored by the U.S. Government under contract No. W91CRB-11-C-0049.

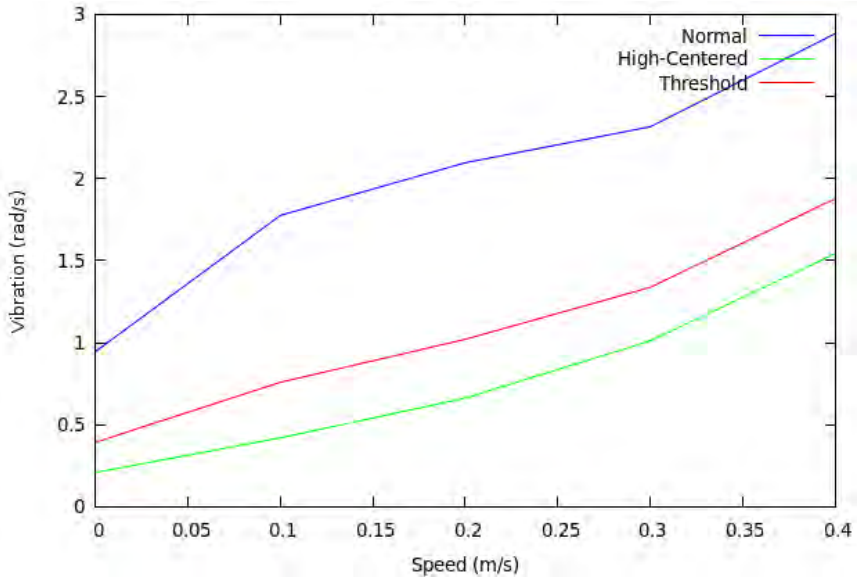

Figure 14: High-centering vibration threshold learned as a function of track speed

A threshold function was then computed as the weighted average of the vibration rate observed during NORMAL and HIGH-CENTERED control regimes:

$$\tau(v) = w\varphi_{high-centered}(v)+(1-w)\varphi_{normal}(v)$$

where *w* is the weight coefficient. To reduce false positive detections of high-centering, we used a value of *w* of 0.75. As a result, the vibration rate must be significantly below the observed vibration level for the NORMAL control regime for a period of time (5 seconds) before the regime classifier determines that the robot is in a HIGH-CENTERED control regime.

Figure 14 shows the interpolated functions for $\varphi_{normal}(v)$, $\varphi_{high-centered}(v)$, and $\tau(v)$ based on learning trials for the robot driving on a flat, carpeted surface. This is a particularly difficult case because the vibration level during motion over flat, smooth terrain is relatively low. Driving over rough terrain would result in higher values of vibration levels in the NORMAL regime, making the task of detecting high-centering easier.

After learning, the high-centering escape behavior acts as a driver-assist capability during teleoperation. The operator drives the robot normally, but if the vibration level remains below threshold for the current commanded velocity for an extended period of time, the regime classifier determines that the robot has become high-centered (Figure 13, left) . At that point, the escape behavior automatically deploys the flippers to attempt to make contact with the ground and commands the robot to move backwards (Figure 13, right). In most cases, this should result in the robot moving back off the obstacle.

6 PackBot Active Weight Shifting

For our fourth task, we developed an MTC-based controller that used DTL to learn how to move the PackBot's manipulator arm to actively shift its center of gravity. The SAM for this task includes a state variable for the robot's pitch angle and an action variable for the rotation angle of the arm's shoulder joint. The anchor point for the NORMAL regime is at a pitch angle of zero and a shoulder angle of zero (arm horizontal). The anchor point for the ROLLOVER regime is at a pitch angle of 90° and a shoulder angle of 90° (arm vertical).

Sponsored by the U.S. Government under contract No. W91CRB-11-C-0049.

The role of the exploratory controller is performed by a human operator. For each trial, the arm is initially positioned with the shoulder joint in the vertical orientation. The operator drives the robot forward at a fixed speed. As the robot climbs the obstacle, it reaches a point at which it tips backward and rolls over. By rotating the shoulder joint, the robot can shift its center-of-gravity forward and avoid rolling over backwards. The learning task is to determine the necessary arm angle to prevent rollover as a function of the robot's pitch angle.

Figure 15: PackBot climbs over obstacle using active weight shifting

For this task, the SAM is represented as an array of values where *SAM(θ, α)* represents the control regime for the robot when its pitch angle is θ and its shoulder angle is α. Initially, all of the array values are set to NORMAL. When the regime classifier identifies that the robot has entered the ROLLOVER regime, DTL updates the SAM for all entries in the history H:

$$\forall(\theta_i, \alpha_i) \in H: \forall\theta \geq \theta_i - \theta_{margin}, \alpha \geq \alpha_i: SAM(\theta, \alpha) = \text{ROLLOVER}$$

where θ_i is the robot's pitch angle at i timesteps prior to the present, α_i is the robot's shoulder angle at i timesteps prior to the present, and θ_{margin} is a safety margin (20° in these experiments). Whenever DTL detects a rollover event, it predicts that rollover will occur again at the same or greater pitch and the same or greater shoulder angle.

As in the rollover prevention task, the DTL acts as a driver-assist behavior for weight shifting. DTL predicts that the robot will rollover when

$$SAM(\theta, \alpha) = ROLLOVER$$

where θ is the current pitch angle and α is the current shoulder angle.

Sponsored by the U.S. Government under contract No. W91CRB-11-C-0049.

When DTL predicts rollover, it overrides the current operator command, stops the robot, and rotates the shoulder joint forward to the maximum angle for which it does not predict ROLLOVER:

$$\alpha_{command} = MAX(\alpha) \mid SAM(\theta, \alpha) = NORMAL$$

where $\alpha_{command}$ is the joint position command that is sent to the robot.

Using this MTC-based controller, the operator can simply command the robot to move forward, and the controller will automatically rotate the arm forward to shift the center-of-gravity and prevent rollover (Figure 15).

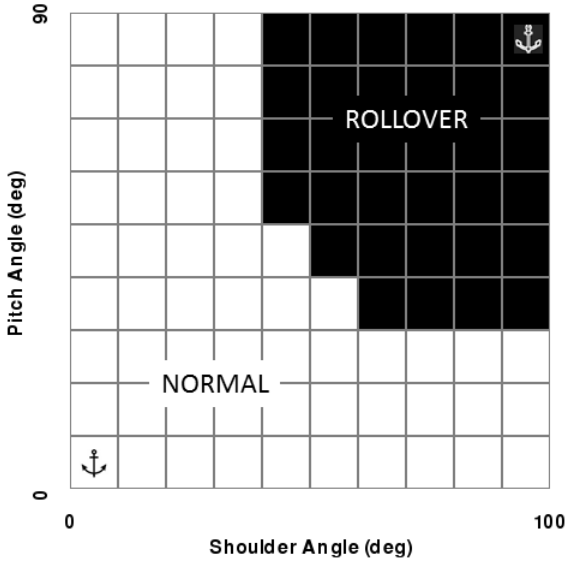

Figure 16: SAM learned by DTL for the weight shifting task. The anchor point for the NORMAL regime is at zero pitch and shoulder angle. The anchor point for the ROLLOVER regime is at the maximum pitch and shoulder angle.

Figure 16 shows a SAM learned by DTL for weight shifting. In this SAM, the pitch and shoulder axes are quantized at 10° intervals, where each grid cell corresponds to a single SAM array value. In our experiments, DTL was able to learn each SAM in 4 trials. A tradeoff exists between the resolution of the SAM grid and the number of trials required. Finer resolutions allow more precise identification of the regime boundary, but require more trials to learn an effective SAM.

7 Related Work

Reinforcement learning techniques [Kaelbling, Littman & Moore 96], such as temporal difference learning [Sutton 88] and Q learning [Watkins 89], have been applied to a wide range of robot control problems, including robot juggling [Schaal & Atkeson 94], obstacle avoidance and corridor following [Smart & Kaelbling 02], quadruped obstacle negotiation [Lee, et al. 06] and gap jumping [Schaal & Atkeson 10], helicopter control [Abbeel, et al. 07], and many others. Most previous research has been done offline or using discrete, rather than continuous state-spaces, but [Schaal & Atkeson 94] and [Abbeel, et al. 07] are examples of real-time control learning in continuous state and action spaces.

Sponsored by the U.S. Government under contract No. W91CRB-11-C-0049.

DTL is similar to reinforcement learning in that it uses reinforcement in the form of undesirable regime detection to learn an appropriate control policy. Like most successful applications of reinforcement learning to robotics, DTL makes use of domain knowledge about the problem to reduce the search space so that learning can occur in a tractable amount of time [Kaelbling, Littman & Moore 96].

DTL differs from previous reinforcement learning algorithms in modeling the state-action space as a set of linear control problems linked by nonlinear transitions. DTL requires additional a priori domain knowledge in terms of the control regimes that may occur and the state-action pairs (anchor points) that are most likely to generate these control regimes. DTL also requires the ability to identify the current control regime based on sensor inputs.

The payoff for these additional requirements is greatly accelerated learning. While many reinforcement learning algorithms require hundreds or thousands of trials and hours of real time to learn robotics tasks, DTL is able to learn highly dynamic control tasks in minutes, often with the number of required trials in the single digits.

However, our initial experiments have been conducted using low-dimensional state-action spaces. As with all learning techniques, DTL will need to address the curse of dimensionality. We believe the DTL will scale well due to the large amount of information that can be gleaned from a single observation and the use of domain knowledge in the form of anchor points.

DTL is also similar to adaptive fuzzy control [Passino & Yurkovich 94] in terms of using heuristic a priori knowledge about the control problem and then modifying the initial control rules based on feedback from the environment. DTL differs both in the specific learning algorithm provided and in the focus on learning behaviors that must deal with abrupt discontinuities in control regimes, such as the transition that occurs from normal driving to snap oversteer.

One limitation of our approach is that the robot must be able to survive the undesirable control regimes experienced during learning or be cheap enough to be easily replaced. In the case of the PackBot learning rollover prevention and weight shifting, it was very useful that we had a rugged platform that could survive multiple falls. In the case of high-speed vehicle control learning with full-sized vehicles, learning may need to occur in simulation first and then be transferred to the actual vehicle for testing.

One way to address this limitation would be to modify the exploratory controller so that it starts with maximally conservative behavior and gradually becomes more aggressive over time. This would reduce the negative effects on the platform, but may require more learning time. We plan to investigate this approach in Phase II of the Dynamo Project.

A second limitation is the lack of an ability to unlearn overly conservative threshold values resulting from false positive regime classifications. A way to address this would be to probabilistically attempt to exceed current thresholds by a small amount. If these attempts do not result in negative control regimes, then the corresponding thresholds would be increased. We plan to explore this approach in Phase II.

8 Public Website

We have launched a public website for the Dynamo Project at:

http://www.irobot.com/dynamo

This website contains a summary of the research performed under the Dynamo Project, along with publications, videos, and source code.

Sponsored by the U.S. Government under contract No. W91CRB-11-C-0049.

9 Phase II Plans

9.1 DTL Scaling and Enhancements

In the Dynamo Feasibility Study, we have introduced MTC, a new framework for developing robot controllers, and DTL, a new algorithm for fast online learning of control regime thresholds. We have demonstrated that DTL can rapidly learn these thresholds in three different task domains: preventing understeer and oversteer for simulated high-speed vehicles, preventing rollover for the PackBot, and enabling the PackBot to use active weight shifting to climb obstacles that it would not be otherwise able to climb.

In Phase II, we plan to study how DTL scales to higher-dimensional state-action spaces. For example, we plan to develop rollover prevention for lateral as well as longitudinal rollover, and to develop weight shifting behaviors in three dimensions to maintain balance over rough terrain. We also plan to study how to implement "unlearning" to correct for noisy data or incorrect regime classifications. In addition, we plan to investigate exploratory behaviors that, instead of beginning with maximally aggressive behavior, start with conservative behavior and gradually become more aggressive.

9.2 High-Speed Small UGV DSC Experiments

We will apply the results from the Feasibility Study to a UGV hardware platform (Figure 17). We plan to modify an existing high-speed, radio-controlled vehicle platform, such as the Traxxas Slash 4x4, for use as the Dynamo vehicle testbed. The Slash 4x4 is a 1/10 scale vehicle (22" long x 11" wide x 5" high) with a top speed of 60 mph. We integrate a computational payload with sensors (GPS, IMU, LIDAR) for vehicle control.

Figure 17: Dynamo high-speed small UGV hardware platform

We will perform experiments to measure the MTC DSC controller's ability to learn the transition boundaries between full grip, understeer, and oversteer on a physical vehicle. A key challenge will be to make the learning algorithm sufficiently robust to deal with uncertainties in velocity (from GPS) and yaw rate (from the IMU) estimates.

9.3 Integrating Body Mass Manipulation with DSC for High-Speed Control

We also plan to investigate the integration of body mass modulation with DSC, using a unique two-dimensional weight shifting payload (Figure 18) that is capable of quickly shifting the

Sponsored by the U.S. Government under contract No. W91CRB-11-C-0049.

vehicle center-of-gravity. This payload will be similar to a gantry robotic system with two linear actuators that can move a weight rapidly in two-dimensional XY space, requiring just 0.5 seconds to move the weight from front to back and 0.2 seconds to move the weight from right to left.

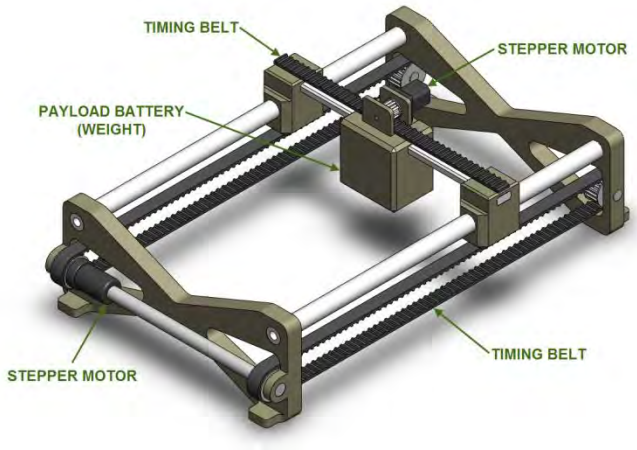

Figure 18: Two-dimensional weight shifting payload

We will extend our MTC-based DSC controller to incorporate states and actions of the weight shifting payload into the state-action space. Initially, we will perform experiments with statically-modeled changes to the vehicle's center-of-gravity. For example, the controller may move the weight to the rear to avoid oversteer, but it will not accelerate and decelerate the weight so quickly that we need to consider the dynamic effects of this motion on vehicle stability. Subsequently, we will examine the dynamic effects of accelerating and decelerating the weight quickly. A key challenge will be to maintain tractability of learning as the dimensionality of the state-action space increases.

9.4 PackBot Adaptive Rough Terrain Mobility

We will extend our research in rough terrain mobility to more complex outdoor environments, such as the terrain in our PackBot Test Facility. The PackBot Test Facility includes rock piles, rubble piles, sand pits, simulated railroad tracks, and a shipping container, as well as other examples of challenging terrain. We will extend our Rollover Prevention and Active Weight Shifting capabilities to deal with complex 3D real-world terrain, including body mass manipulation to maintain stability along both pitch and roll axes.

We will also investigate how to adapt in real-time to changes in terrain, such as moving from rocky terrain to sandy terrain. This will incorporate DTL extensions such as unlearning and progression from conservative to aggressive behavior in the exploratory controller.

10 References

[Abbeel, et al. 07] P. Abbeel, A. Coates, M. Quigley, and A. Y. Ng, "An application of reinforcement learning to aerobatic helicopter flight," in *Advances in Neural Information Processing Systems 19: Proc. 20st Annual Conf. (NIPS 2006)*, B. Scholkopf, J. Platt, and T. Hofmann, Eds., Cambridge, MA: MIT Press, 2007, pp. 1-8.

[Jackowski 10] Jeffery Jackowski, Deputy Project Manager, Robotic Systems Joint Project Office, RSJPO PM Priority List Submitted to JGRE, January 2010.

Sponsored by the U.S. Government under contract No. W91CRB-11-C-0049.

[Kaelbling, Littman & Moore 96] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research,* vol. 4, pp. 237-285, 1996.

[Lee, et al. 06] H. Lee, Y. Shen, C.-H. Yu, G. Singh, and A. Ng, "Quadruped robot obstacle negotiation via reinforcement learning," in *Proc. 2006 IEEE Int. Conf. on Robotics and Automation,* Orlando, FL, 2006, pp. 3003-3010.

[Passino & Yurkovich 94] K. Passino and S. Yurkovich, *Fuzzy Control,* Menlo Park, CA: Addison-Wesley Longman, 1994.

[Schaal & Atkeson 94] S. Schaal and C. G. Atkeson, "Robot juggling: An implementation of memory-based learning," *Control Systems Magazine*, vol. 14, no. 1, pp. 57-71, 1994.

[Schaal & Atkeson 10] S. Schaal and C. G. Atkeson, "Learning control in robotics," *IEEE Robotics & Automation Magazine,* vol. 17, no. 2, pp. 20-29, 2010.

[Smart & Kaelbling 02] W. D. Smart and L. P. Kaelbling, "Effective reinforcement learning for mobile robots," in *Proc. 2002 IEEE Int. Conf. on Robotics and Automation*, Washington, DC, 2002, pp. 3404-3410.

[Sutton 88] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine Learning*, vol. 3, pp. 9-44, 1988.

[Watkins 89] C. Watkins, "Learning from delayed rewards," Ph.D. dissertation, University of Cambridge, Cambridge, UK, 1989.

Sponsored by the U.S. Government under contract No. W91CRB-11-C-0049.

11 Appendix: Abbreviations

ABS	Anti-lock Braking System
DNF	Did Not Finish
DTL	Dynamic Threshold Learning
DSC	Dynamic Stability Control
EOD	Explosives Ordnance Disposal
IED	Improvised Explosive Device
MRAS	Model Reference Adaptive System
MTC	Model Transition Control
N/A	Not Applicable
PID	Proportional-Integral-Derivative
RSJPO	Robotic Systems Joint Projects Office
SAM	State-Action Map
STR	Self-Tuning Regulator
TORCS	The Open Race Car Simulator
UGV	Unmanned Ground Vehicle

Sponsored by the U.S. Government under contract No. W91CRB-11-C-0049.